1. Flipping an Image

// time complexity: O(n^2)

// space complexity: O(1)

class Solution {

public int[][] flipAndInvertImage(int[][] image) {

//1. flip the image horizontally

for(int i = 0; i<image.length; i++){

int j = 0;

int k = image[i].length - 1;

while(j < k){

int temp = image[i][j];

image[i][j++] = image[i][k];

image[i][k--] = temp;

}

//2. inversion of the image

for(j = 0; j<image[i].length; j++){

image[i][j] = image[i][j] == 1 ? 0:1;

}

}

return image;

}

}

2. Transpose of a Matrix

// time complexity: O(n^2)

// space complexity: O(1)

class Solution {

public int[][] transpose(int[][] A) {

int R = A.length, C = A[0].length;

int[][] ans = new int[C][R];

for (int r = 0; r < R; r++)

for (int c = 0; c < C; c++) {

ans[c][r] = A[r][c];

}

return ans;

}

}

3. Search a 2D Matrix

// time complexity: O(log(m \* n))

// space complexity: O(1)

class Solution {

public boolean searchMatrix(int[][] matrix, int target) {

// number of rows

int m = matrix.length;

// base case condition

if(m == 0){

return false;

}

// number of columns

int n = matrix[0].length;

// binary search algorithm

int low = 0, high = m \* n - 1;

int midIdx, midElement, rowIdx, colIdx;

while(low <= high){

midIdx = low + (high - low)/2;

rowIdx = midIdx / n;

colIdx = midIdx % n;

midElement = matrix[rowIdx][colIdx];

// condition 1 - match

if(target == midElement){

return true;

}

else{

if(target < midElement){

// condition 2 - left side

high = midIdx - 1;

}

else{

// condition 3 - right side

low = midIdx + 1;

}

}

}

return false;

}

}

4. Set Matrix Zeroes

Brute Force Approach

// time complexity: O(m \* n)

// space complexity: O(m +n)

class Solution {

public void setZeroes(int[][] matrix) {

int m = matrix.length;

int n = matrix[0].length;

Set<Integer> rows = new HashSet<Integer>();

Set<Integer> cols = new HashSet<Integer>();

for(int i=0; i<m; i++){

for(int j=0; j<n; j++){

if(matrix[i][j] == 0){

rows.add(i);

cols.add(j);

}

}

}

for(int i=0; i<m; i++){

for(int j=0; j<n; j++){

if(rows.contains(i) || cols.contains(j)){

matrix[i][j] = 0;

}

}

}

}

}